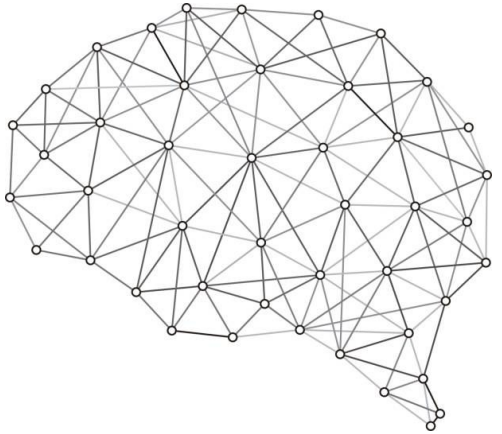


Aula 15-Modelo Cliente Servidor - HTTP e CoAP

Instituto Iracema
PESQUISA E INOVAÇÃO

Módulo de Internet das Coisas

- Profª. Nídia Glória da Silva Campos

UNIVERSIDADE ESTADUAL DO CEARÁ

MCTI FUTURO

Softex

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO

IA

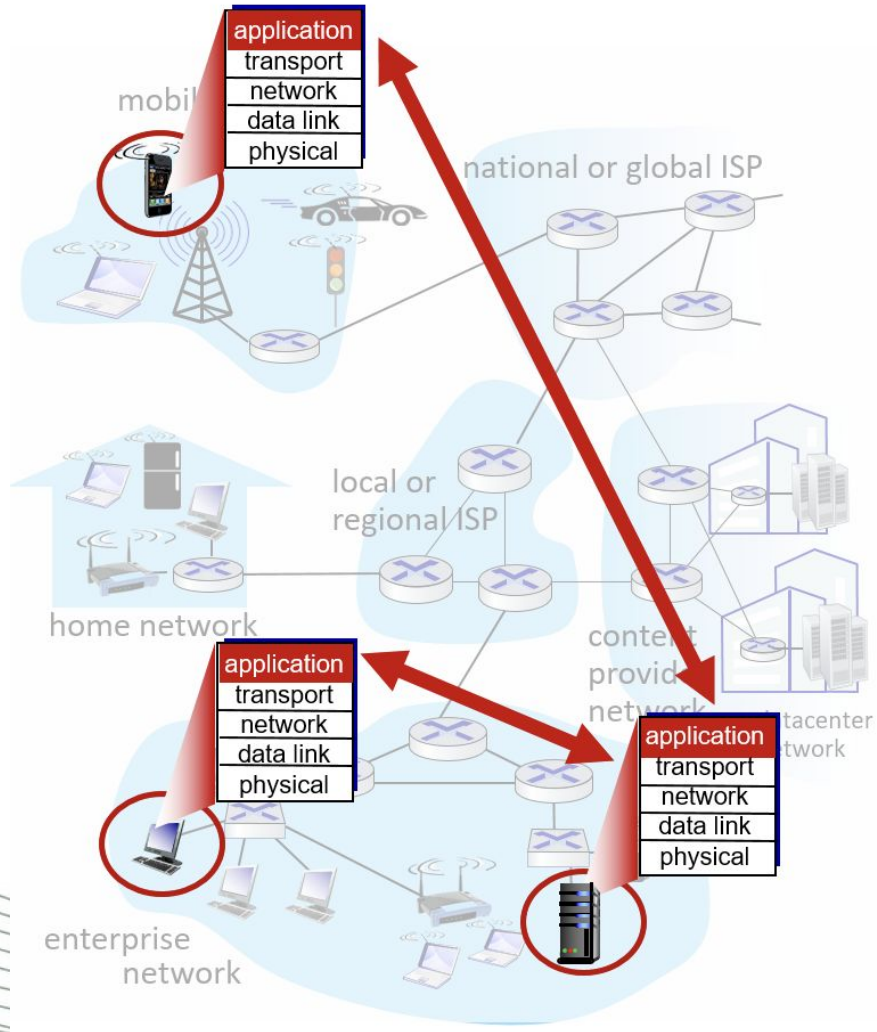
Objetivos da Aula

- Apresentar os modelos de aplicação de rede
- Conhecer os principais requisitos de aplicações de redes
- Conhecer detalhes dos protocolos HTTP e CoAP

IA

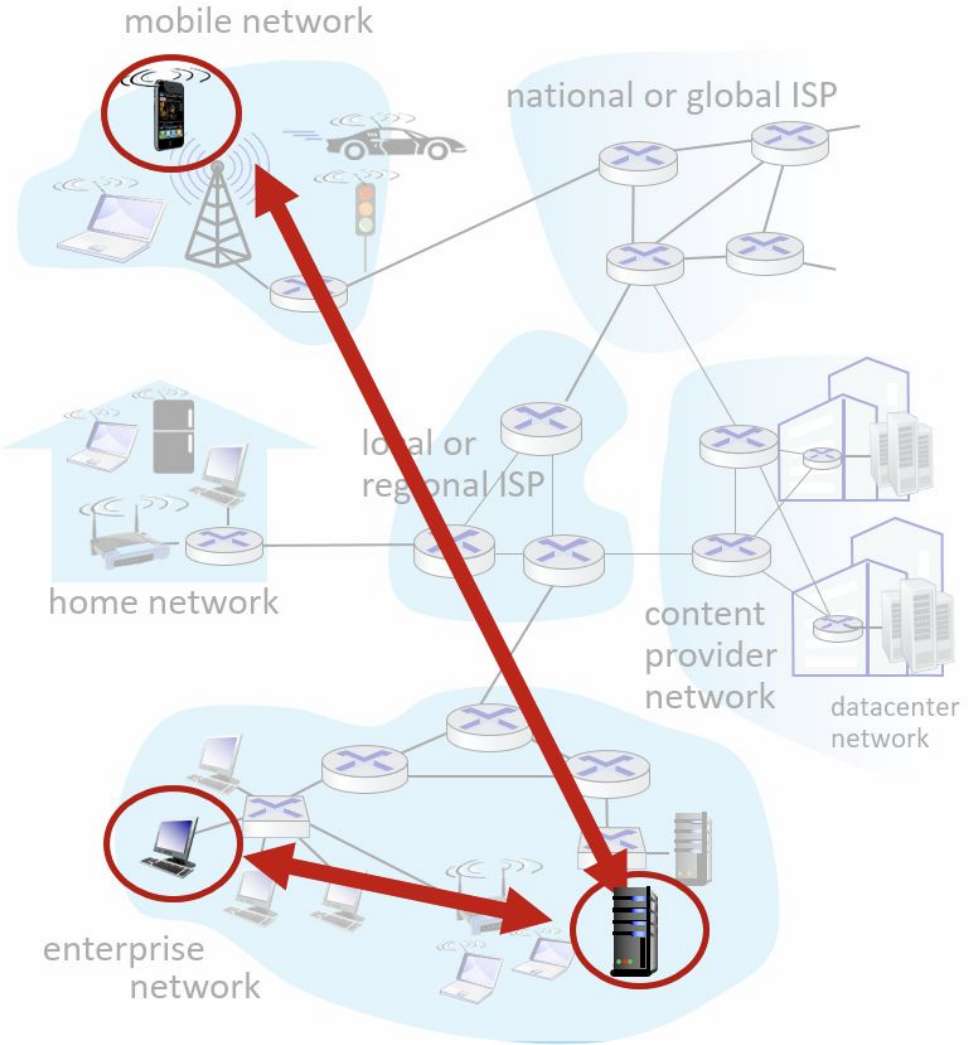
Criando uma aplicação de rede [1]

- **criar programas que:**
  - executem em diferentes sistemas finais
  - comunique-se sobre a rede
  - ex. servidor se comunicando com um navegador web
- **sem necessidade de escrever código para o núcleo da rede**
  - dispositivos de núcleo não executam aplicações de sistemas finais
  - aplicações nos sistemas finais o rápido desenvolvimento e propagação das apps.



Paradigma do Cliente-Servidor [1]

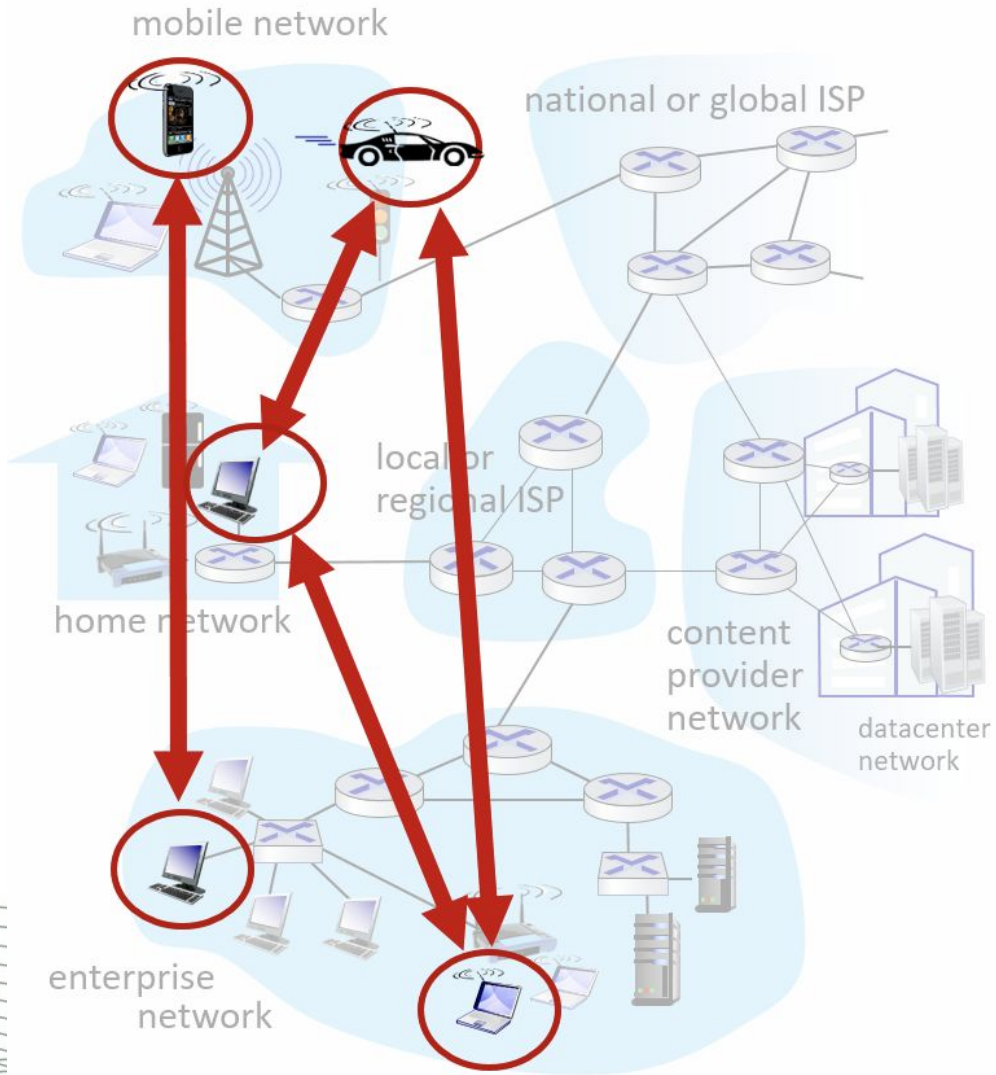
- servidor
  - sempre executando no host
  - endereço IP permanente
  - muitas vezes em data centers, para escalabilidade
- cliente
  - contacta, se comunica com o servidor
  - pode estar conectado de modo intermitente
  - pode ter endereços IP dinâmicos
  - não se comunicam diretamente entre si
  - ex. HTTP, FTP, IMAP



IA

Arquitetura Peer-to-Peer [1]

- sem servidor sempre conectado
- sistemas finais arbitrários se comunicam diretamente entre si
- peers requisitam serviços de outros peers, provêm serviços para outros peers
  - **alta escalabilidade:** novos peers trazem mais capacidade de serviço, como também novas demandas de serviço
- peers estão conectados de maneira intermitente e mudam de endereço IP
  - gerenciamento complexo
- ex. compartilhamento de arquivo P2P



IA

Comunicação de processos [1]

- **processo**: programa executando em um host
- dentro do mesmo host, dois processos podem se comunicar usando **comunicação inter-processo** (provida pelo SO)
- processos em diferentes hosts se comunicam usando **mensagens**

clientes e servidores

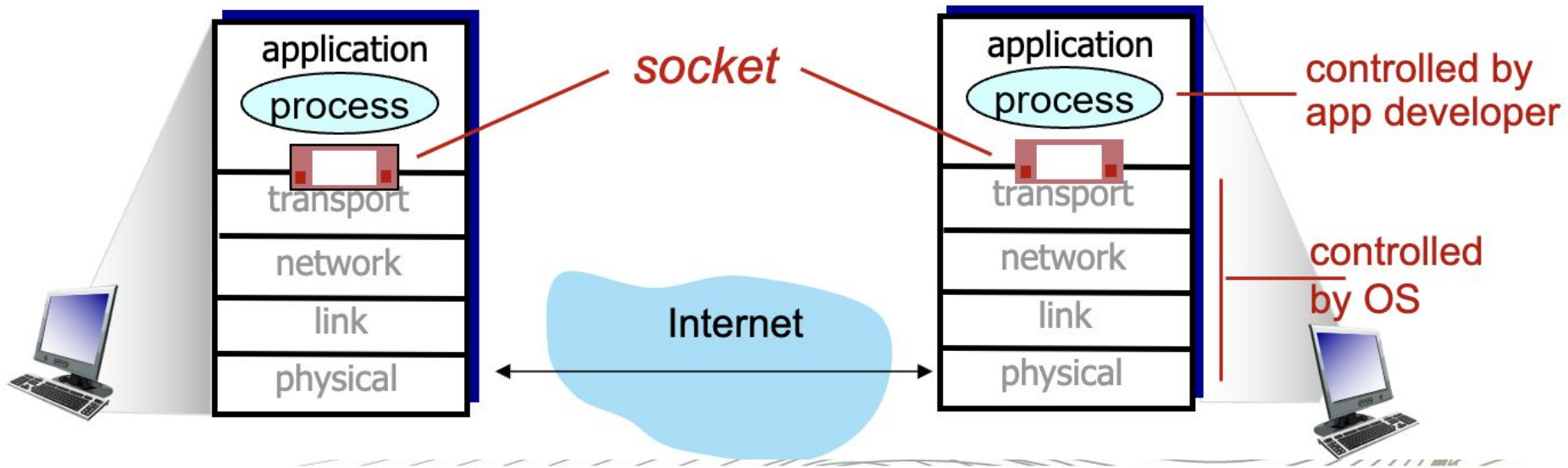
- **processo cliente**: processo que inicia a comunicação
- **processo servidor**: que espera ser contactado

nota: aplicações de arquitetura P2P tem processo cliente e processo servidor

IA

Sockets [1]

- processos enviam e recebem mensagens através de seu socket
- socket é análogo a uma porta
  - o processo emissor empurra a mensagem porta a fora
  - o processo emissor confia na infraestrutura de transporte do outro lado da porta para entregar a mensagem ao socket do processo receptor
  - dois sockets envolvidos: um de cada lado da comunicação



IA

Endereçamento de processos [1]

- para receber mensagens, um processo deve ter um **identificador**
- um host possui um único endereço IP IPv4 de 32 bits
- **P.** *O endereço IP do host onde o processo executa é suficiente para identificar o processo de rede?*
  - **R.** Não, podemos ter vários processos executando no host
- o **identificador** deve ter tanto o **endereço IP** quanto o **número de porta** associados ao processo no host
- exemplo de números de porta
  - servidor HTTP: 80
  - servidor SMTP: 25
- para mandar uma mensagem HTTP para o servidor web gaia.sc.umass.edu:
  - **IP:** 129.113.245.12
  - **número de porta:** 80

IA

Os protocolos da camada de aplicação definem: [1]

- **tipos de mensagens trocadas**, ex. request, response
- **sintaxe da mensagem**
  - quais campos estão na mensagem e como são delineados
- **semântica da mensagem**
  - o significado da informação nos campos
- **regras** para quando e como os processos enviam e respondem mensagens

protocolos abertos:

- definidos em RFCs, todos podem acessar a definição do protocolos
- permite a interoperabilidade
- ex. HTTP, SMTP

protocolos proprietários:

- ex. Skype, Zoom

IA Qual serviço da camada de transporte uma aplicação precisa? [1]

integridade de dados

- algumas aplicações precisam que 100% dos dados sejam entregues (ex. transferência de arquivo)
- outras toleram alguma perda de dados (ex. áudio stream)

tempo

- algumas aplicações precisam de baixo atraso para serem efetivas (ex. VoIP, games online)

vazão

- algumas aplicações precisam de uma vazão mínima para serem efetivas (ex. multimídia)
- outras podem usar qualquer quantidade (ex. elásticas)

segurança

- criptografia, integridade de dados,...

IA

Requisitos do serviço de transporte: apps comuns [1]

aplicação	perda dados	vazão	sensível ao tempo
Transf. Arqu./download	sem perda	elática	não
e-mail	sem perda	elástica	não
Páginas Web	sem perda	elástica	não
Áudio/video tempo-real	tolerante	audio: 5Kbps-1Mbps video:10Kbps-5Mbps	sim, 10's mseg
streaming áudio/video	tolerante	O mesmo acima	sim, poucos seg.
Games interativos	tolerante	Kbps+	sim, 10's mseg
Mensagens de texto	sem perda	elástica	sim e não

IA

Serviços dos protocolos de transporte da Internet [1]

Serviço TCP

- ***transporte confiável*** entre processo emissor e receptor
- ***controle de fluxo:*** emissor não sobrecarrega o receptor
- ***controle de congestionamento:*** emissor diminui a vazão quando a rede está sobrecarregada
- ***orientado a conexão:*** configuração necessária entre processos cliente e servidor
- ***não provê:*** temporização, vazão mínima, segurança

Serviço UDP

- ***transferência de dados não-confiável*** entre processos emissor e receptor
- ***não provê:*** confiabilidade, controle de fluxo, controle de congestionamento, temporização, garantia de vazão, segurança e estabelecimento de conexão

IA

Aplicações da Internet, protocolos de transporte [1]

aplicação	Protocolo da camada de aplicação	protocolo de transporte
Transf. Arq./download	FTP [RFC 959]	TCP
e-mail	SMTP [RFC 5321]	TCP
documentos Web	HTTP 1.1 [RFC 7320]	TCP
Telefonia na Internet	SIP [RFC 3261], RTP [RFC 3550], ou proprietário	TCP ou UDP
streaming audio/video	HTTP [RFC 7320], DASH	TCP
Games interativos	WOW, FPS (proprietário)	UDP ou TCP

IA

Segurança do TCP [1]

Sockets TCP e UDP

- sem criptografia
- senhas em texto claro enviadas pelo socket atravessa a Internet!

Transport Layer Security (TLS)

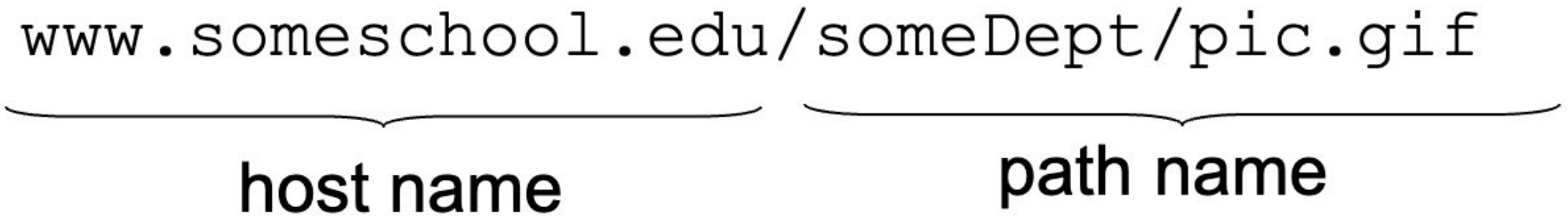
- provê conexões TCP criptografadas
- integridade de dados
- autenticação ponta-a-ponta

TLS é implementado na camada de aplicação

- apps usam bibliotecas TLS, que usam o TCP por sua vez
- texto claro é enviado no socket e atravessa a Internet ***criptografado***

A Web e o HTTP [1]

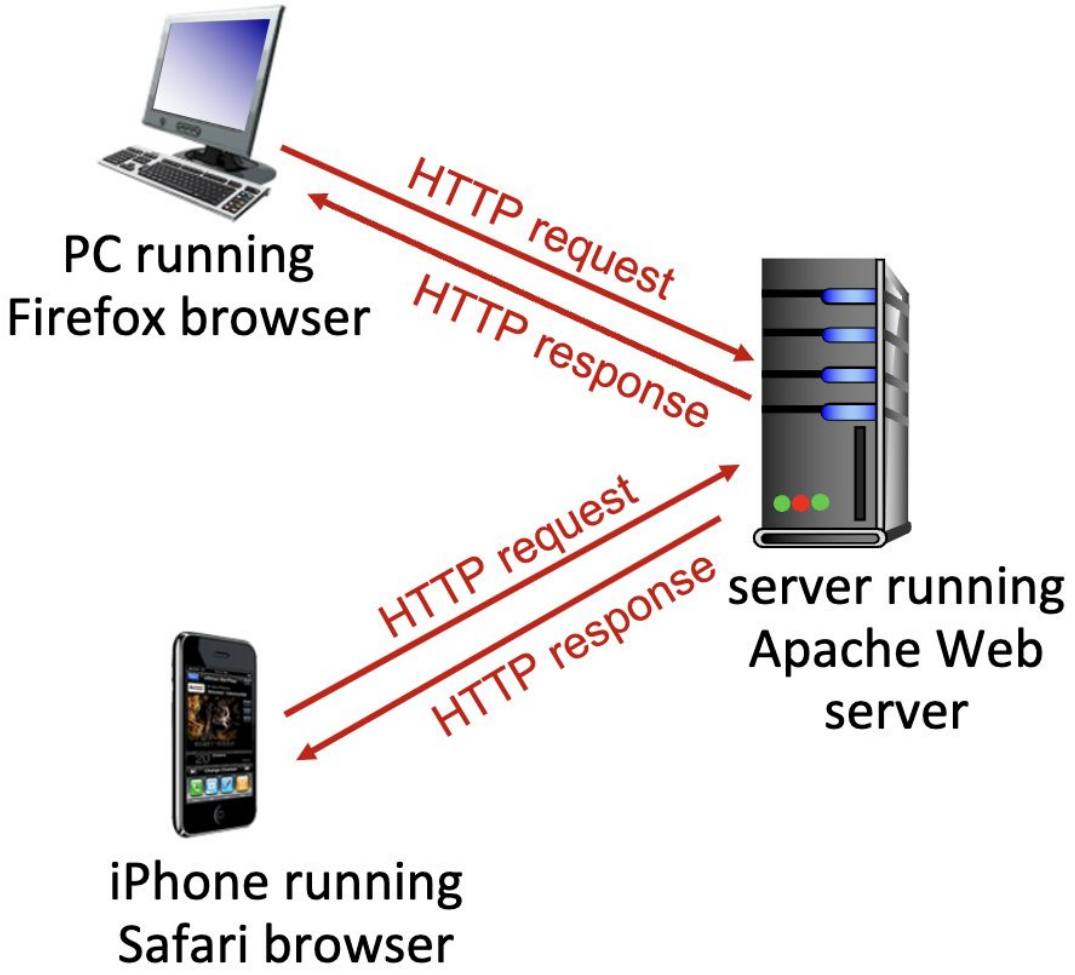
- uma página web consiste em ***objetos***, cada qual pode ser armazenado em diferentes servidores Web
- um objeto pode ser arquivo HTML, imagem JPEG, arquivo de áudio,...
- uma página web consiste no ***arquivo HTML base*** que inclui ***vários objetos referenciados***, cada qual por uma ***URL***, ex.



Visão Geral do HTTP [1]

HTTP: hypertext transfer protocol

- protocolo da camada de aplicação
- modelo cliente/servidor
  - ***cliente***: navegador que requisita, recebe (usando o protocolo HTTP) e mostra objetos Web
  - ***servidor***: servidor Web envia (usando o protocolo HTTP) objetos em resposta às requisições



IA

Visão Geral do HTTP [1]

HTTP usa o TCP

- cliente inicia a conexão TCP (cria socket) com o servidor, porta 80
- servidor aceita a conexão TCP do cliente
- mensagens HTTP são trocadas entre o navegador e servidor web
- conexão TCP é encerrada

HTTP é sem estado

- o servidor não mantém informação sobre as requisições passadas do cliente

IA

Conexões HTTP: dois tipos [1]

Não-Persistente

1. a conexão TCP é aberta
2. somente um objeto é enviado pela conexão TCP
3. a conexão TCP é fechada

download de vários objetos precisa de várias conexões

HTTP Persistente

- uma conexão TCP é aberta para um servidor
- vários objetos podem ser enviados em uma única conexão TCP entre o cliente e o servidor
- a conexão TCP é fechada

Conexão Não-Persistente: exemplo [1]

Usuário digita: **`www.someSchool.edu/someDepartment/home.index`**
(contém texto, referências para 10 imagens jpeg)

1a. Cliente HTTP inicia a conexão TCP para o servidor HTTP (processo) em www.someSchool.edu na porta 80

1b. Servidor HTTP no host www.someSchool.edu esperando por conexão TCP na porta 80 "aceita" conexão, notifica cliente

2. cliente HTTP envia a *mensagem request HTTP* (contendo a URL) no socket TCP da conexão. Mensagem indica que o cliente quer o objeto someDepartment/home.index

3. Servidor HTTP recebe a mensagem request, forma *mensagem response* contendo o objeto requsitado, e envoa a mensagem no seu socket

time

Conexão Não-Persistente: exemplo [1]

Usuário digita: `www.someSchool.edu/someDepartment/home.index`

(contém texto, referencia 10 imagens jpeg)

4. Servidor HTTP fecha a conexão TCP.

5. Cliente HTTP recebe a mensagem de resposta contendo o arquivo html, mostra o html. Renderizando o html encontra 10 imagens jpeg referenciadas.

6. Passos 1-5 são repetidos para cada uma dos 1o objetos jpeg

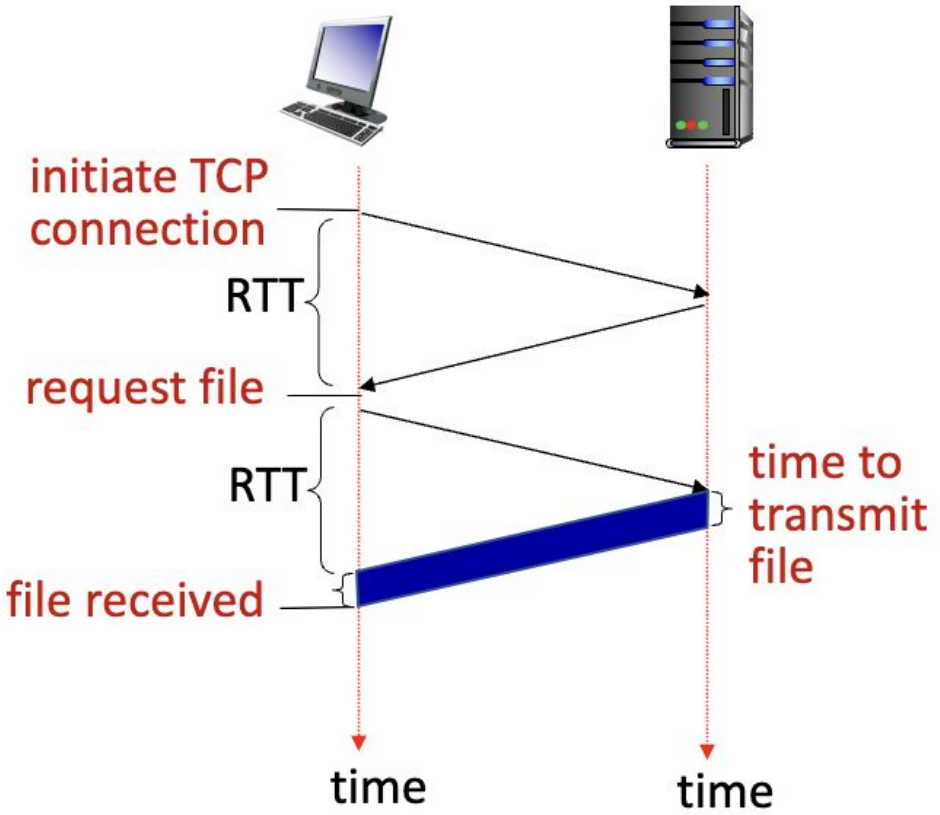
time

HTTP Não-Persistente: tempo de resposta [1]

RTT (definição): tempo para que um pequeno pacote viaje do cliente para o servidor e retorne

tempo de resposta HTTP (por objeto):

- um RTT para iniciar a conexão TCP
- um RTT para requisição HTTP e os primeiros bytes da resposta HTTP chegarem
- tempo de transmissão do arquivo/objeto



tempo de resposta da conexão HTTP não-persistente = 2RTT + tempo transmissão do arquivo

IA

HTTP Persistente (HTTP 1.1) [1]

Problemas do HTTP Não-Persistente

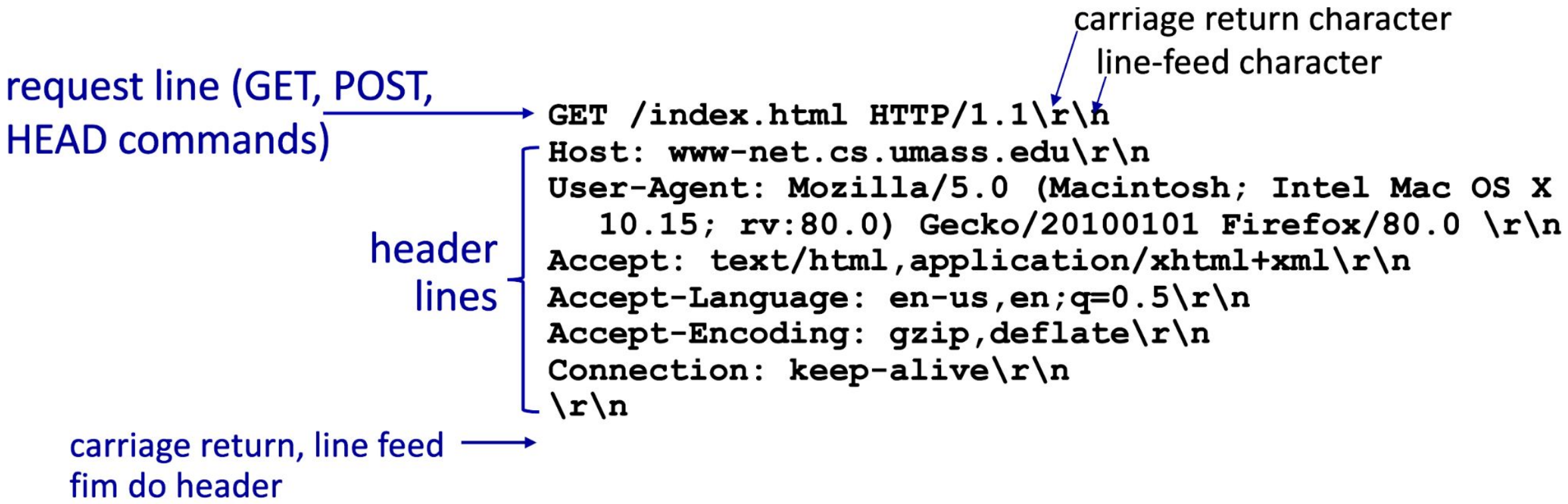
- requer 2 RTTs por objeto
- sobrecarrega o SO para cada conexão TCP
- navegadores geralmente abrem várias conexões TCP paralelas para buscar objetos referenciados em paralelo

HTTP Persistente

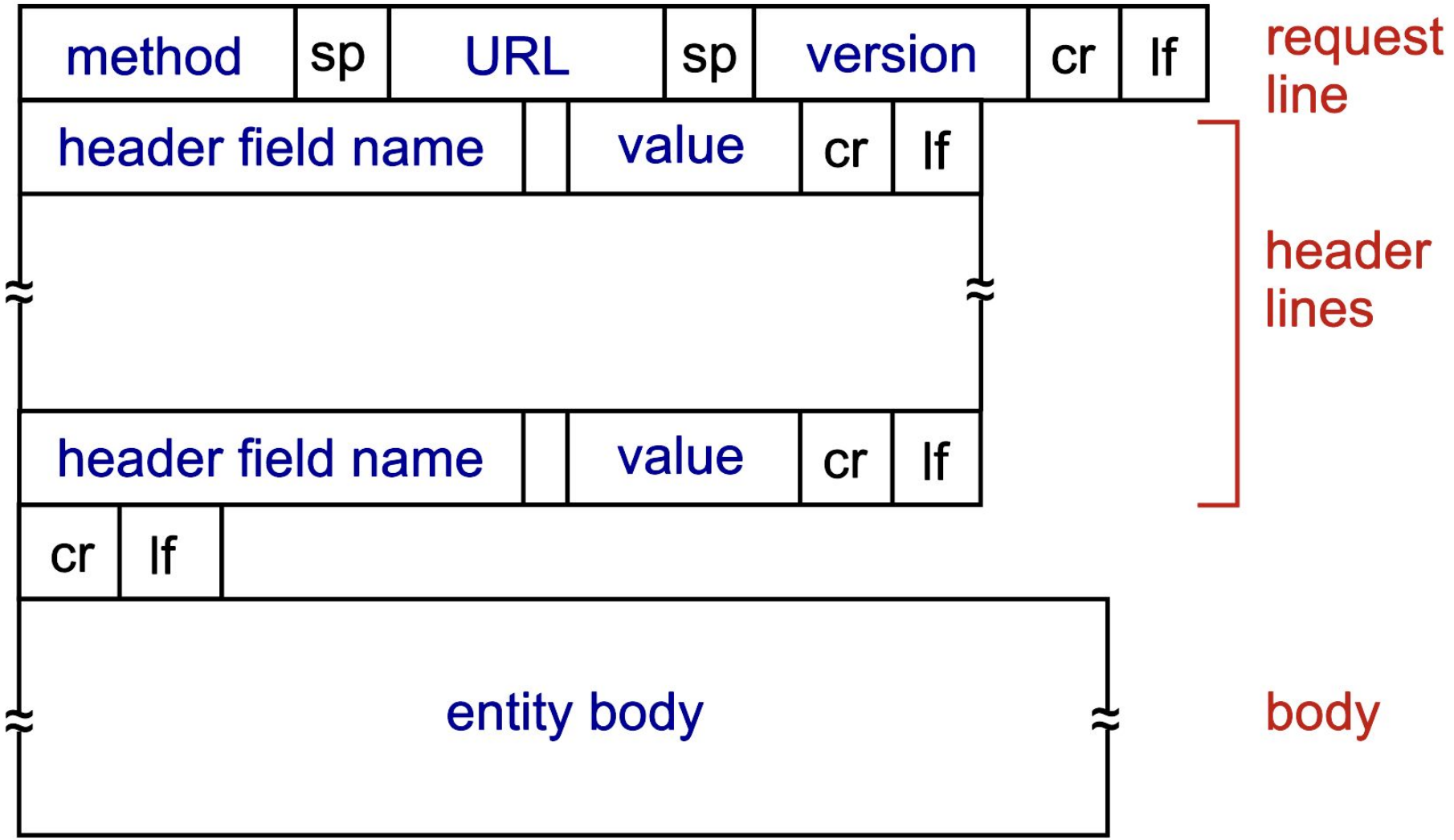
- o servidor deixa a conexão aberta depois de enviar a resposta
- mensagens HTTP subsequentes entre o mesmo cliente e servidor enviadas sobre a conexão aberta
- cliente envia requisições tão logo ele encontra um objeto referenciado
- apenas 1 RTT para todos os objetos referenciados (cortando pela metade o tempo de resposta)

Mensagem HTTP Request [1]

- dois tipos de mensagens: request, response
- Mensagem HTTP Request
  - codificada em ASCII (formato legível)



Mensagem HTTP Request: formato geral [1]



IA

Outras mensagens HTTP Request [1]

Método POST:

- página web muitas vezes inclui entradas de formulário
- a entrada do usuário envia pelo cliente ao servidor no corpo da entidade da mensagem request POST HTTP

Método GET (para enviar dados ao servidor):

- inclui dados do usuário no campo da URL da mensagem request GET HTTP (seguido por um '?')

```
www.somesite.com/animalsearch?monkeys&banana
```

Método HEAD:

- requisita headers (somente) que podem ser retornados se a URL especificada foi requisitada com um método GET HTTP

Método PUT:

- upload de um novo arquivo (objeto) ao servidor
- substitui o arquivo que existe na URL especificada com o conteúdo no corpo da entidade da mensagem HTTP POST`

IA

Mensagem HTTP Response [1]

status line (protocol status code status phrase)

header lines

data, e.g., requested HTML file

```
HTTP/1.1 200 OK
Date: Tue, 08 Sep 2020 00:53:20 GMT
Server: Apache/2.4.6 (CentOS)
   OpenSSL/1.0.2k-fips PHP/7.4.9
   mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Tue, 01 Mar 2016 18:57:50 GMT
ETag: "a5b-52d015789ee9e"
Accept-Ranges: bytes
Content-Length: 2651
Content-Type: text/html; charset=UTF-8
\r\n
data data data data data ...
```

IA

Códigos de estatus da mensagem HTTP Response

- o código de estatus aparece na primeira linha da mensagem de resposta do servidor para o cliente
- alguns exemplos:

  200 OK

- requisição bem-sucedida, objeto requisitado nesta mensagem de resposta

  301 Moved Permanently

- objeto requisitado movido, nova localização especificada mais adiante nesta mensagem no campo Location.

  400 Bad Request

- mensagem de requisição não foi compreendida pelo servidor

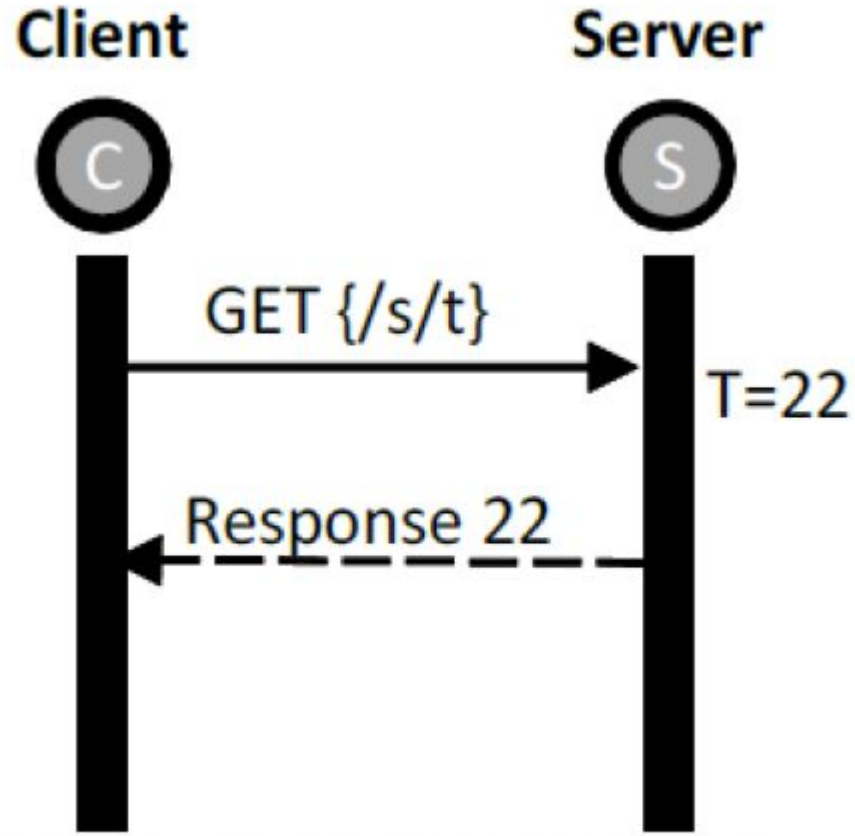
  404 Not Found

- documento requisitado não foi encontrado no servidor

  505 HTTP Version Not Supported

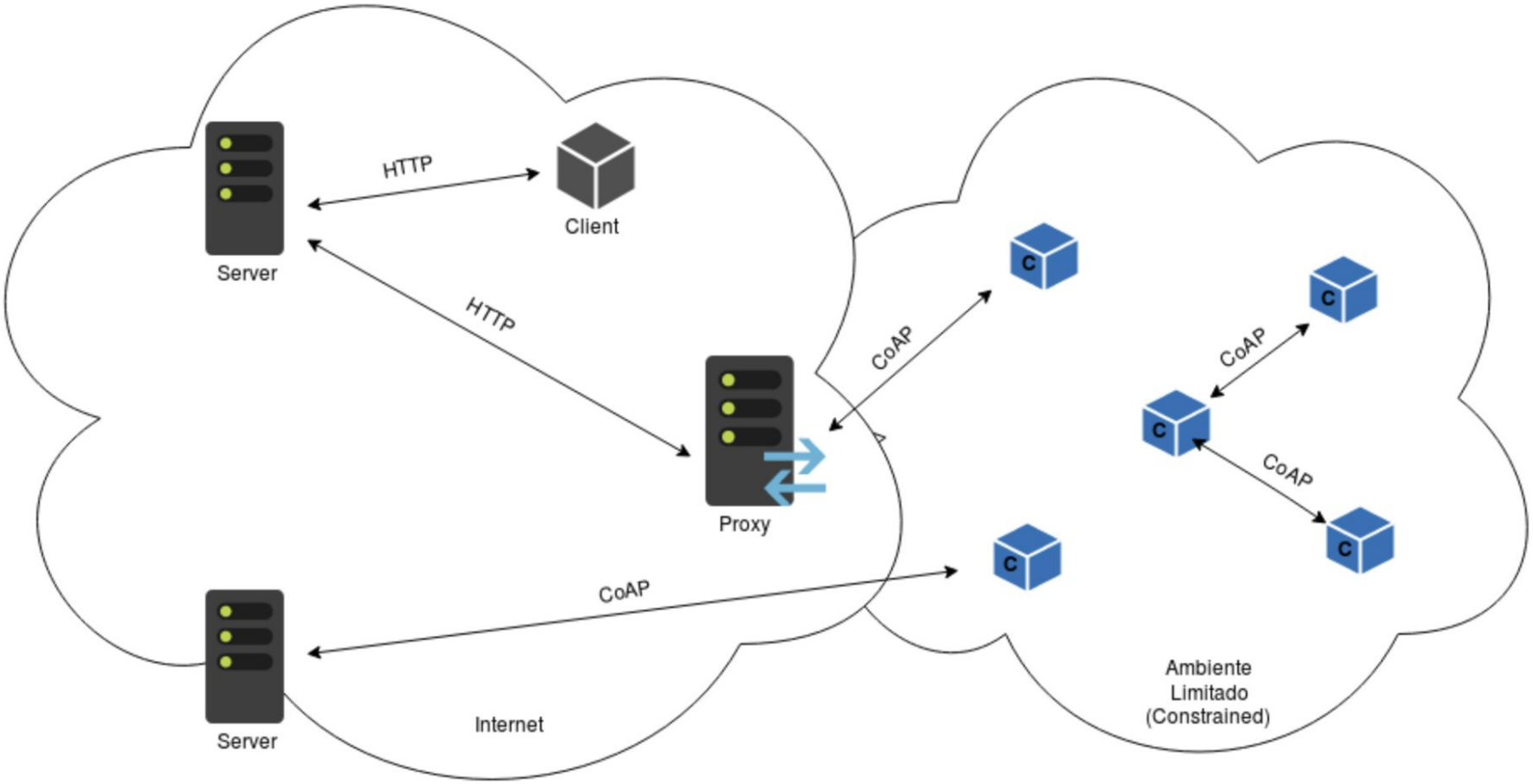
CoAP - Constrained Application Protocol [2]

- Padronizado pelo IETF (RFC 7252)
- Arquitetura de aplicação Cliente-Servidor
- Usa padrão REST (REpresentational State Transfer)
  - Cada recurso é identificado por um URI (Uniform Resource Identification)
- Baixo overhead: 4 bytes
- Usados em redes com restrições de recursos: memória, processamento, baixa largura de banda e altas taxas de erros (ex. 6LowWPAN)



CoAP - Mapeamento do HTTP para o CoAP [4]

- quando dispositivos não conseguem operar o CoAP
- dispositivos inteligentes compartilham informações na Web



IA

CoAP - Modelos de Mensagens [3]

- Executa sobre o UDP
  - Sem criptografia: UDP/5683
  - Com criptografia (DTLS): UDP/5684

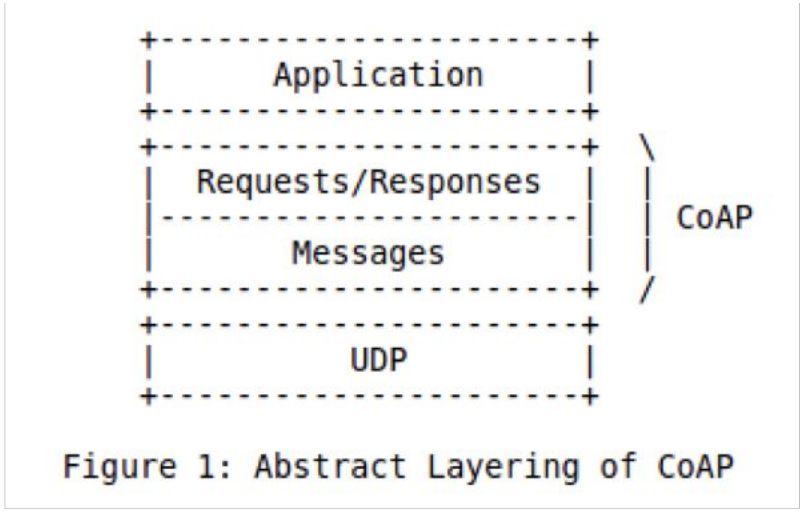


Figure 1: Abstract Layering of CoAP

- Possui mecanismos de confiabilidade de entrega de dados:
  - Flags CON (Confirmable) e ACK (Acknowledgement)

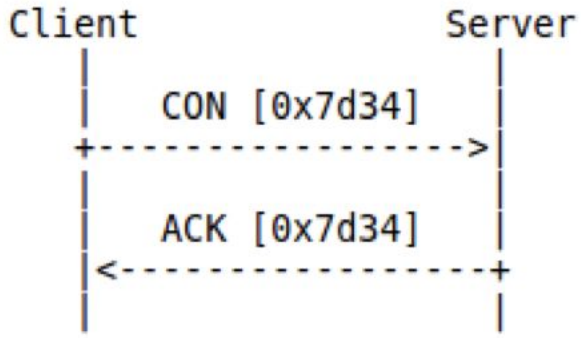


Figure 2: Reliable Message Transmission

CoAP - Modelos de Mensagens [3]

- Funciona sem confiabilidade de entrega de dados: flag NON

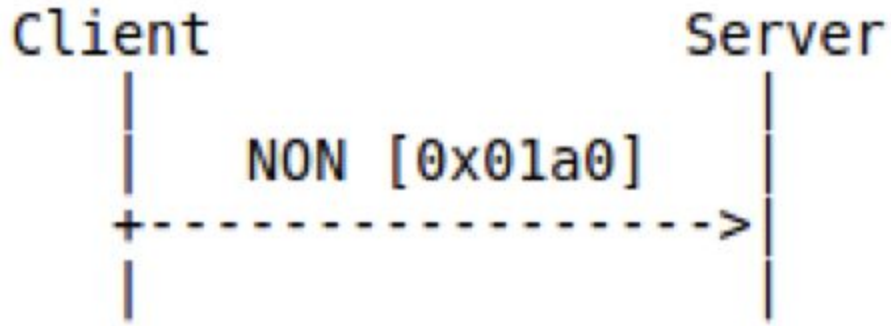
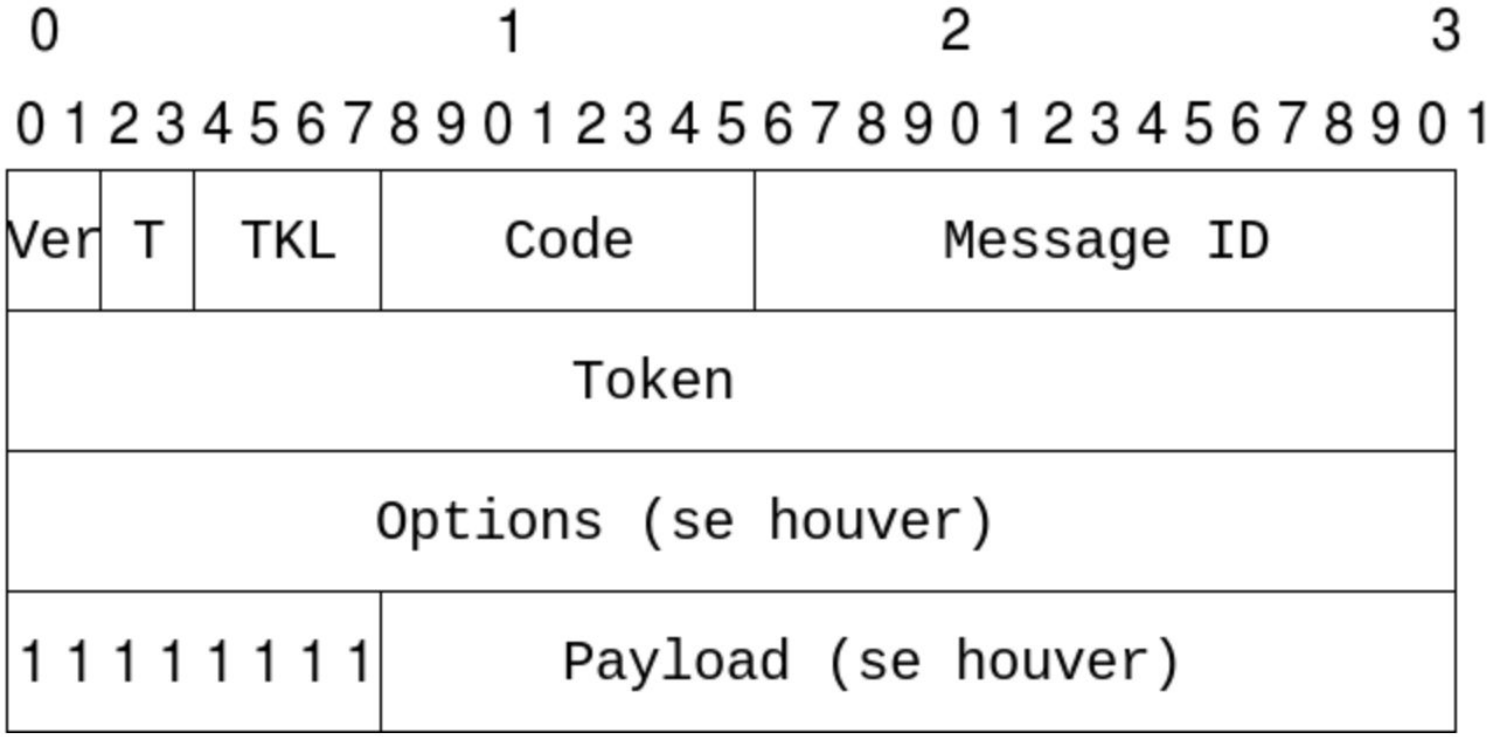


Figure 3: Unreliable Message Transmission

CoAP - Formato da Mensagem [4]

- Tipo (T): Indica se a mensagem é de um dos tipos:
  - CON: Confirmável (0)
  - NON: Não-confirmável (1)
  - ACK: Confirmação (2)
  - RST: Reset (3)



- Comprimento de Token (TKL) :

IA

CoAP - Modelo Request/Response [3]

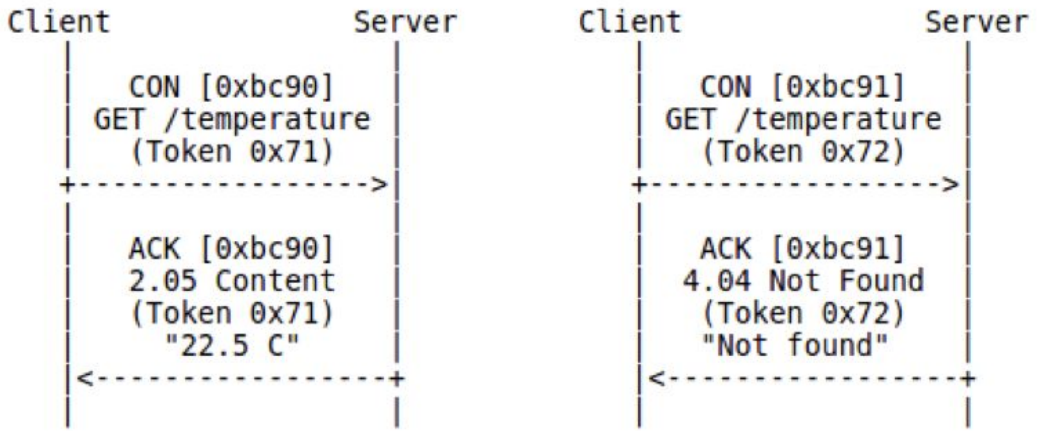


Figure 4: Two GET Requests with Piggybacked Responses

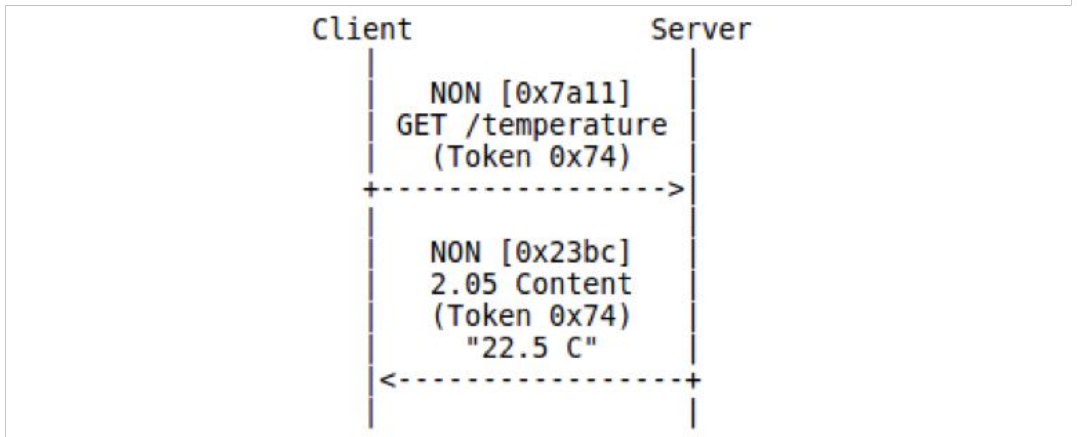


Figure 6: A Request and a Response Carried in Non-confirmable Messages

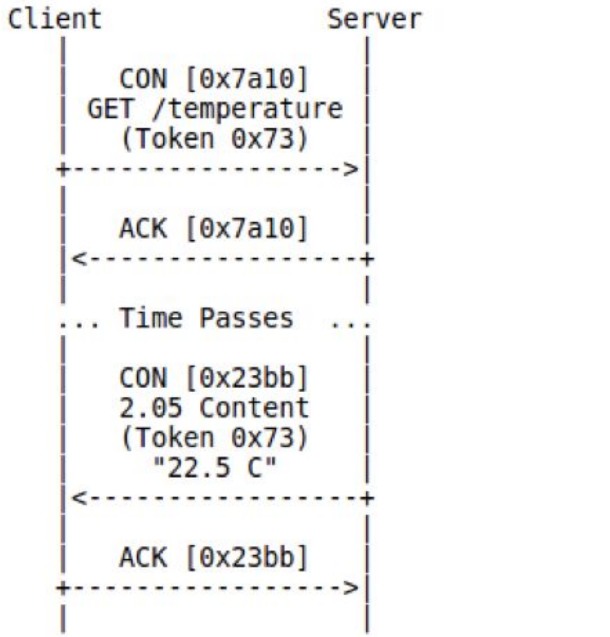


Figure 5: A GET Request with a Separate Response

- Message ID = combina msgs de controle (CON, ACK)
- Token = “request ID”

IA

CoAP - Semântica das Requisições e Respostas [3]

Método das requisições

- **GET**
  - solicita recursos do servidor
- **POST:**
  - cria ou atualiza recursos no servidor
  - ações definidas pelo servidor
- **PUT:**
  - cria e atualiza recursos no servidor com tipo de mídia ou conteúdo apropriado
- **DELETE:** apagar recursos

Códigos e frases de Respostas

- **2.01 Created**
- **2.02 Deleted**
- **2.03 Valid**
- **2.04 Changed**
- **2.05 Content**

IA

Referências Bibliográficas

[1] J. Kurose, K. Ross. Redes de Computadores e a Internet - Uma abordagem top-down. 8ª ed. Pearson. 2020.

[2] G. K. Teklemariam, J. Hoebeke, F. Van den Abeele, I. Moerman and P. Demeester, "Simple RESTful sensor application development model using CoAP," 39th Annual IEEE Conference on Local Computer Networks Workshops, Edmonton, AB, Canada, 2014, pp. 552-556, doi: 10.1109/LCNW.2014.6927702.

[3]Z. Shelby; K. Hartke; C. Bormann. The Constrained Application Protocol (CoAP). Disponível em: <https://tools.ietf.org/html/rfc7252>. Acesso em: 23/03/2023.

[4] A. da Silva; F. Carvalho; M. Nazário. Constrained Application Protocol (CoAp). 2019. Disponível em: <https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/coap/>. Acesso em: 23/03/2023.

Dúvidas?

Módulo de Internet das Coisas

Universidade Estadual do Ceará
Instituto Iracema
Pesquisa e Inovação
MCTI Futuro
Softex
Ministério da Ciência, Tecnologia e Inovação
Governo Federal
Brasil
União e Reconstrução